Alexandria University

Faculty of Engineering

Computer and Systems Engineering

First Year

CS 111: Probability Theory

Fall 2021

Lab 1

Overview

R is a programming language and software environment for statistical analysis, graphics representation and reporting. We will be using R in the probability labs to conduct experiments and evaluate results. In this lab, you are required to install R and step up its environment as described, then discover R libraries, tools and packages. To assure a complete understanding of R functions, you are required to execute the program described at the end.

Installation

1. R Installation

Please follow the instructions suitable for the Operating System of your Laptop

- **Windows**
  1. Open an internet browser and go to https://cloud.r-project.org/
  2. Click on the "Download R for Windows" link at the top of the page.
  3. Click on the "install R for the first time" link at the top of the page.
  4. Click "Download R for Windows" and save the executable file somewhere on your computer. Run the .exe file and follow the installation instructions.
  5. Now that R is installed, you need to download and install RStudio.
- **Linux**

  open your terminal and run the following commands:

```
sudo apt update
sudo apt install r-base
sudo apt build-dep r-base
```

2. RStudio Installation

- **Windows**
    1. Go to www.rstudio.com and click on the "Download RStudio" button.
    2. Click on "Download RStudio Desktop."
    3. Click on the version recommended for your system, or the latest Windows version, and save the executable file. Run the .exe file and follow the installation instructions.
- **Linux**

Open the terminal and run the following commands:

```
1. $ sudo apt -y install wget
   https://download1.rstudio.org/desktop/bionic/amd64/rstudio-1
   .2.5001-amd64.deb
2. $ wget
   https://download1.rstudio.org/desktop/bionic/amd64/rstudio-1
   .2.5001-amd64.deb
3. $ sudo dpkg -i rstudio-1.2.1578-amd64.deb
```

If you encounter any dependency problems, run:

```
$ sudo apt -f install
```

To start RStudio, run the following command in the terminal:

```
$ rstudio
```

R Language

Data Types

Data Type	Example	Verify

Logical	TRUE, FALSE	`v <- TRUE`
		`print(class(v))`
		it produces the following result
		`[1] "logical"`

Numeric	12.3, 5, 999	`v <- 23.5`
		`print(class(v))`
		it produces the following result
		`[1] "numeric"`

Integer	2L, 34L, 0L	`v <- 2L`
		`print(class(v))`
		it produces the following result
		`[1] "integer"`

Complex	3 + 2i	`v <- 2+5i`
		`print(class(v))`
		it produces the following result
		`[1] "complex"`

Character	'a' , '"good", "TRUE", '23.4'	`v <- "TRUE"`
		`print(class(v))`
		it produces the following result
		`[1] "character"`

Raw	"Hello" is stored as 48 65 6c 6c 6f	`v <- charToRaw("Hello")`
		`print(class(v))`
		it produces the following result
		`[1] "raw"`

Vector	['bleu', 'red', 'green']	`# Create a vector.`
		`apple <- c('red','green',"yellow")`
		`print(apple)`
		`# Get the class of the vector.`
		`print(class(apple))`
		it produces the following result
		`[1] "red"    "green"  "yellow"`
		`[1] "character"`

List	[[1]]
	[1] 2 5 3
	[[2]]
	[1] 21.3
	[[3]]
	function (x) .Primitive("sin")
		`# Create a list.`
		`list1 <- list(c(2,5,3),21.3,sin)`
		`# Print the list.`
		`print(list1)`
		Try it yourself !

Matrix	[,1] [,2] [,3]
	[1,] "a" "a" "b"
	[2,] "c" "b" "a"
		`# Create a matrix.`
		`M = matrix(`
		`c('a','a','b','c','b','a'), nrow = 2,`
		`ncol = 3, byrow = TRUE)`
		`print(M)`
		it produces the following result
		`     [,1] [,2] [,3]`
		`[1,] "a"  "a"  "b"`
		`[2,] "c"  "b"  "a"`

Array	, , 1
	[,1] [,2] [,3]
	[1,] "green" "yellow" "green"
	[2,] "yellow" "green" "yellow"
	[3,] "green" "yellow" "green"
	, , 2
	[,1] [,2] [,3]
	[1,] "yellow" "green" "yellow"
	[2,] "green" "yellow" "green"
	[3,] "yellow" "green" "yellow"
		`# Create an array.`
		`a <- array(c('green','yellow'),dim =`
		`c(3,3,2))`
		`print(a)`
		
		Try it yourself !

Factor	[1] green green yellow red red red green
	
	Levels: green red yellow
	
	[1] 3
		`# Create a vector.`
		`apple_colors <-`
		`c('green','green','yellow','red','red','red','green')`
		
		`# Create a factor object.`
		`factor_apple <- factor(apple_colors)`
		
		`# Print the factor.`
		`print(factor_apple)`
		`print(nlevels(factor_apple))`
		
		Try it yourself !

Data Frame	gender height weight Age
	
	1 Male 152.0 81 42
	
	2 Male 171.5 93 38
	
	3 Female 165.0 78 26
		`# Create the data frame.`
		`BMI <-    data.frame(`
		`   gender = c("Male",`
		`"Male","Female"),`
		`   height = c(152, 171.5, 165),`
		`   weight = c(81,93, 78),`
		`   Age = c(42,38,26)`
		`)`
		`print(BMI)`
		
		Try it yourself !

Variables

A variable in R can store an atomic vector, group of atomic vectors or a combination of many Robjects. A valid variable name consists of letters, numbers and the dot or underline characters. The variable name starts with a letter or the dot not followed by a number.

Variable Assignment

The variables can be assigned values using leftward, rightward and equal to operator. The values of the variables can be printed using print() or cat() function. The cat() function combines multiple items into a continuous print output

```
# Assignment using equal operator.

var.1 = c(0,1,2,3)

# Assignment using leftward operator.

var.2 <- c("learn","R")

# Assignment using rightward operator.

c(TRUE,1) -> var.3
```

Data Type of a Variable

In R, a variable itself is not declared of any data type, rather it gets the data type of the R - object assigned to it. So R is called a dynamically typed language, which means that we can change a variable's data type of the same variable again and again when using it in a program.

```
var_x <- "Hello"

var_x <- 34.5
```

Finding Variables

To know all the variables currently available in the workspace we use the ls() function. Also the ls() function can use patterns to match the variable names.

```
print(ls())

# List the variables starting with the pattern "var".

print(ls(pattern = "var"))
```

Deleting Variables

Variables can be deleted by using the rm() function

```
rm(var.3)

print(var.3)

[1] "var.3"

Error in print(var.3) : object 'var.3' not found
```

Functions

Built-in

refer to the mostly wided used R functions in this link
ex: sum(), length(), unique(), max(), which () ...

User-defined

We can create user-defined functions in R. They are specific to what a user wants and once created they can be used like the built-in functions. Below is an example of how a function is created and used.

```r
# Create a function with arguments.
new.function <- function(a,b,c) {
   result <- a * b + c
   print(result)
}
# Call the function by position of arguments.
new.function(5,3,11)
# Call the function by names of the arguments.
new.function(a = 11, b = 5, c = 3)
```

LOOPS

Repeat loop

Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.

While loop

Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.

For loop

Like a while statement, except that it tests the condition at the end of the loop body.

Decision Making

1	**if statement** An if statement consists of a Boolean expression followed by one or more statements.
2	**if...else statement** An if statement can be followed by an optional else statement, which executes when the Boolean expression is false.
3	**switch statement** A switch statement allows a variable to be tested for equality against a list of values.

Application

For a better understanding of R functions, you are required to write a program that does the following:

1. Create a vector to store these numbers : 12, 7, 3, 4.2, 18, 2, 54, -21, 8, -5
2. Compute the mean using 2 different methods.
3. Find the median [1] value.
4. Find the mode [2] value.

Notes:

[1] median is the middle value in a set of data.

[2] mode is the value that has the highest number of occurrences in a set of data.

***Hints**:*

[1] No built-in function exists for mode in R. You'll need to create your own version.

*[2] **tabulate()** function takes the integer-valued vector bin and counts the number of times each integer occurs in it.*

*[3] **match (x, table)** returns a vector of the positions of (first) matches of its first argument in its second where x is your input vector and table are the values to be matched against*

References

1. https://www.tutorialspoint.com/r/index.htm
2. https://cloud.r-project.org
3. https://rstudio.com